WHAT IS CLAIMED IS:

1. A computer-implemented method of dynamically loading protocol stacks, comprising:

receiving a message to load a first protocol stack;

determining whether the first protocol stack can be loaded;

unloading a second protocol stack if the first protocol stack cannot be initially loaded; and

loading the first protocol stack.

2. The method of claim 1, wherein the first protocol stack cannot be initially loaded because memory was not available to load the first protocol stack.

3. The method of claim 1, wherein the first protocol stack cannot be initially loaded because the first and second protocol stacks are not compatible.

4. The method of claim 1, further comprising accessing a database for procedures for loading the first protocol stack.

5. The method of claim 1, further comprising accessing a database for procedures for unloading the second protocol stack.

6. The method of claim 3, further comprising accessing a database to determine that the first and second protocol stacks are not compatible.

7. The method of claim 1, wherein the first protocol is loaded by launching a process or launching a service.

8. The method of claim 1, wherein the second protocol is unloaded by sending a message to a process, killing a process or stopping a service.

9. The method of claim 1, wherein the message specifies portions of the first protocol stack that are to be loaded.

10. A computer program product that dynamically loads protocol stacks, comprising:

computer code that receives a message to load a first protocol stack;

computer code that determines whether the first protocol stack can be loaded;

computer code that unloads a second protocol stack if the first protocol stack cannot be initially loaded;

computer code that loads the first protocol stack; and

a computer readable medium that stores the computer codes.

11. The computer program product of claim 11, wherein the computer readable medium is a CD-ROM, floppy disk, tape, flash memory, system memory, hard drive, or a data signal embodied in a carrier wave.

12. A system, comprising:

a processor; and

a computer readable medium storing a computer program including computer code that receives a message to load a first protocol stack, computer code that determines whether the first protocol stack can be loaded, computer code that unloads a second protocol stack if the first protocol stack cannot be initially loaded, and computer code that loads the first protocol stack.

13. The system of claim 12, wherein the computer readable medium is a CD-ROM, floppy disk, tape, flash memory, system memory, hard drive, or a data signal embodied in a carrier wave.

14. A computer-implemented method of dynamically loading protocol stacks, comprising:

a first node sending a message to a second node to load a first protocol stack;

the second node receiving the message to load the first protocol stack;

the second node determining whether the first protocol stack can be loaded;

the second node unloading a second protocol stack if the first protocol stack cannot be initially loaded on the second node; and

the second node loading the first protocol stack.

15. The method of claim 14, wherein the first and second nodes are in different devices.

16. The method of claim 14, wherein the first and second nodes are the same node in a single device.

17. The method of claim 14, wherein the first protocol stack cannot be initially loaded on the second node because memory was not available to load the first protocol stack.

18. The method of claim 14, wherein the first protocol stack cannot be initially loaded on the second node because the first and second protocol stacks are not compatible.

19. The method of claim 14, further comprising accessing a database for procedures for loading the first protocol stack on the second node.

20. The method of claim 14, further comprising accessing a database for procedures for unloading the second protocol stack on the second node.

21. The method of claim 18, further comprising accessing a database to determine that the first and second protocol stacks are not compatible.

22. The method of claim 14, wherein the first protocol is loaded on the second node by launching a process or launching a service.

23. The method of claim 14, wherein the second protocol is unloaded on the second node by sending a message to a process, killing a process or stopping a service.

24. The method of claim 14, wherein the message specifies portions of the first protocol stack that are to be loaded on the second node.

25. A computer program product that dynamically loads protocol stacks, comprising:

computer code that sends a message from a first node to a second node to load a first protocol stack;

computer code that receiving the message to load the first protocol stack on the second node;

computer code that determines whether the first protocol stack can be loaded on the second node;

computer code that unloads a second protocol stack on the second node if the first protocol stack cannot be initially loaded on the second node;

computer code that loads the first protocol stack on the second node; and

a computer readable medium that stores the computer codes.

26. The computer program product of claim 25, wherein the computer readable medium is a CD-ROM, floppy disk, tape, flash memory, system memory, hard drive, or a data signal embodied in a carrier wave.

27. A system, comprising:

a first node that sends a message to a second node to load a first protocol stack; and

the second node that receives the message to load the first protocol stack, determines whether the first protocol stack can be loaded, unloads a second protocol stack if the first protocol stack cannot be initially loaded on the second node, and loads the first protocol stack.

28. The system of claim 27, wherein the computer readable medium is a CD-ROM, floppy disk, tape, flash memory, system memory, hard drive, or a data signal embodied in a carrier wave.